

MA8020 Tekniska beräkningar

Något om felanalys och datoraritmetik

Mikael Hindgren



HÖGSKOLAN
I HALMSTAD

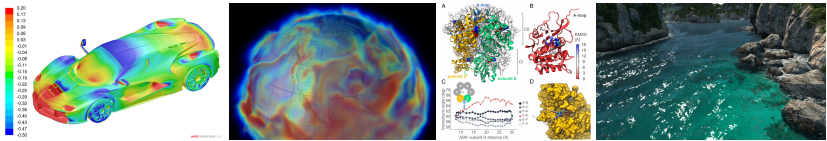
6 november 2025

Numeriska beräkningar

Varför? Man vill matematiskt modellera fysikaliska problem som t.ex.

- Strömningsmekanik inom bil- och flygindustrin
- Klimat- och vädtersimuleringar
- Molekylberäkningar inom bioteknik och läkemedelsdesign
- Animationer inom film- och spelindustrin

Modellerna blir snabbt för komplicerade för att kunna lösas analytiskt!



Fördelar jämfört med verkliga experiment:

- Kräver ingen dyr experimentell utrustning
- Vid utveckling kan prototyputvecklingsfasen minskas
- Man kan modellera/simulera processer som inte går att utföra experimentellt

OBS! En fysikalisk/matematisk modell är alltid en förenklad bild av verkligheten!

Numeriska metoder och numerisk analys

Numerisk metod:

- En algoritm (= följd av regler) som löser ett kontinuerligt (oändligt) matematiskt problem med ett ändligt antal aritmetiska operationer (+, −, ·, /)
- Ger en approximativ lösning

Numerisk analys:

- Konstruktionen av numeriska metoder genom att ersätta matematikens oändliga processer med ändliga processer
- Analys av metoderna med avseende på noggrannhet

Numeriska metoder och numerisk analys

Numeriska beräkningar är aldrig exakta: Felanalys är centralt!

- Före beräkningen:
 - Vilken noggrannhet krävs?
 - Avvägning: Beräkningstid vs felstorlek
 - Välj metod som ger lagom stort fel
- Efter beräkningen:
 - Felets storlek: Vilken noggrannhet (antal gällande siffror) har utdata?

Numeriska beräkningar/simuleringar:

- 1 Förenkla problemet (approximera) så att det kan lösas med dator
- 2 Analysera lösningsmetoden för att bestämma felet i resultatet
- 3 Välj parametrar i lösningsmetoden för att nå önskad noggrannhet och beräkningstid

Felkällor vid numeriska beräkningar

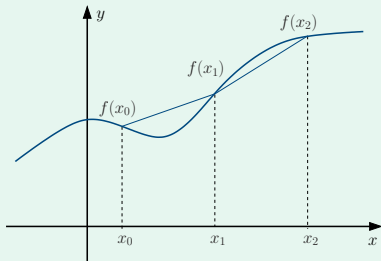
Exempel 1

Vi vill beräkna en integral med hjälp av en tabell med funktionsvärden:

$$I = \int_0^1 f(x) dx$$

| x | 0.20 | 0.40 | 0.60 | 0.80 | 1.00 |
|--------|------|------|------|------|------|
| $f(x)$ | 1.75 | 1.92 | 1.56 | 1.36 | 0.85 |

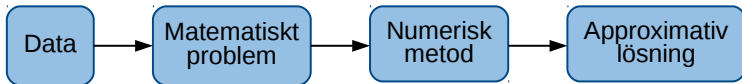
Trapetsmetoden: Approximera $f(x)$ styckvis med räta linjer och beräkna arean under linjerna.



$$I = \int_{x_0}^{x_2} f(x) dx \approx A = h \left(\frac{f(x_0)}{2} + f(x_1) + \frac{f(x_2)}{2} \right), \quad h = x_i - x_{i-1}$$

Vilka felkällor finns?

Felkällor vid numeriska beräkningar



Definition 1

Resultatets olika felkällor klassificeras enligt följande:

- R_X : **Fel i indata**:
 - R_{XF} : Fel som orsakats av felaktiga funktionsvärden
 - R_{XX} : Fel som orsakats av andra fel i indata (t.ex. om indata baseras på en modell)
- R_B : **Avrundningsfel** pga ändligt, fixt antal siffror
- R_T : **Trunkeringsfel** pga att "oändlig process ersätts med ändlig"

Totala felets storlek:

$$|R_{\text{Tot}}| \leq |R_{XF}| + |R_{XX}| + |R_B| + |R_T|$$

Det är viktigt att kunna uppskatta hur stora felen i en beräkning är!

Felkällor vid numeriska beräkningar

Exempel 1 (forts)

Resultatet blir

$$I \approx A = 0.2 \left(\frac{1.75}{2} + 1.92 + 1.56 + 1.36 + \frac{0.85}{2} \right) = 1.228 \approx 1.2$$

Felkällor:

- Funktionen $f(x)$ approximeras med rätta linjer. Detta ger ett trunkeringsfel

$$|R_T| = |I - A|$$

Felet bör bero på $f(x)$ och på steglängden h .

- Felaktiga tabellvärden $f(x_k)$ ger ett fel R_{XF} .
- Funktionsvärdena kan t.ex. vara baserade på en modell av verkligheten vilket ger ett fel R_{XX}
- Slutavrundning av svaret ger ett fel R_B .

Hur ska vi uppskatta felens storlek?

Felkällor vid numeriska beräkningar

Definition 2

Låt x^* beteckna exakta värdet och x ett närmevärde till x^* :

- **Absolut fel:** $\epsilon_a = |\Delta x| = |x^* - x|$
- **Relativt fel:** $\epsilon_r = \frac{|\Delta x|}{|x^*|} \approx \frac{|\Delta x|}{|x|} \leftarrow$ om inte x^* är känd
- Om $\epsilon_a \leq 0.5 \cdot 10^{-t}$ så har x t **korrekta decimaler**
- Heltalssiffrorna (förutom inledande 0:or) och de korrekta decimalerna är **signifikanta siffror**

Exempel 2

$$\begin{aligned}
 x^* &= \sqrt{2} = 1.41421356... \approx x = 1.41 \\
 \Rightarrow |\Delta x| &= 0.00421356... < 0.005 = 0.5 \cdot 10^{-2}
 \end{aligned}$$

$\therefore x = 1.41$ har 2 korrekta decimaler och 3 signifikanta siffror.

Felkällor vid numeriska beräkningar

Mer om totalt fel

Antag att

- $f^*(x^*)$ beskriver ett fenomen exakt (det man vill beräkna)
- f är funktionen som beräknas i datorn och x är mätvärdet (approximationer till f^* resp. x^*)

$$\Rightarrow R_{\text{Tot}} = f(x) - f^*(x^*) = (f(x) - f^*(x)) + (f^*(x) - f^*(x^*)) = e_B + e_D$$

- **Beräkningsfel** $e_B = R_B + R_T$ uppstår då datorn evaluerar värdet.
- **Datafel** $e_D = R_X$ uppstår pga fel i mätinstrument.

Beräkningsalgoritmen påverkar inte datafelet!

Felfortplantning

Problem: Vi vill beräkna $f(x^*)$ då vi endast känner ett närmevärde x . Hur skall vi uppskatta felet

$$|\Delta f| = |f(x^*) - f(x)| ?$$

Sats 1 (Medelvärdessatsen)

Om $f(x)$ är deriverbar finns det ett tal ξ mellan x^* och x sådant att

$$f(x^*) - f(x) = f'(\xi)(x^* - x) = f'(\xi)\Delta x.$$

Om Δx är litet är $f'(\xi) \approx f'(x)$ dvs

$$|\Delta f| \lesssim |f'(x)||\Delta x| \quad \leftarrow \text{Maximalfelsuppskattningen}$$

Exempel 3

Vi vill beräkna $f(x) = \sqrt{x}$ då $x = 3.05 \pm 0.01$

$$\begin{aligned} |\Delta f| \lesssim |f'(x)||\Delta x| &= \frac{1}{2\sqrt{x}}|\Delta x| \leq \frac{0.01}{2\sqrt{3.05}} = 0.002862991... \leq 0.003 \\ \Rightarrow \sqrt{3.05} &= 1.746 \pm 0.003 \end{aligned}$$

Felfortplantning

Resultatet kan generaliseras till flera variabler:

Definition 3

Om $f(x_1, x_2, \dots, x_n)$ är deriverbar och om endast närmevärden

$$x_k = x_k^* + \Delta x_k, \quad k = 1, 2, \dots, n,$$

är kända så är **Maximalfelsuppskattningen**:

$$|\Delta f| \lesssim \left| \frac{\partial f}{\partial x_1} \right| |\Delta x_1| + \dots + \left| \frac{\partial f}{\partial x_n} \right| |\Delta x_n|$$

Anm: Maximalfelsuppskattningen ger en mycket pessimistisk feluppskattning om antalet variabler är stort. Det finns oftast lämpligare sätt att uppskatta felet i sådana fall.

Felfortplantning vid aritmetriska operationer

Sats 2

Låt x_1 och x_2 vara närmevärden till x_1^* och x_2^* . Då gäller:

- $y = x_1 \pm x_2 \Rightarrow |\Delta y| \leq |\Delta x_1| + |\Delta x_2|$
- $y = x_1 \cdot x_2$ eller $y = \frac{x_1}{x_2} \Rightarrow \left| \frac{\Delta y}{y} \right| \leq \left| \frac{\Delta x_1}{x_1} \right| + \left| \frac{\Delta x_2}{x_2} \right|$

Exempel 4

Med $x_1 = 101 \pm 1$ och $x_2 = 100 \pm 1$ får vi:

$$y = x_1 + x_2 = 201, \quad |\Delta y| \leq |\Delta x_1| + |\Delta x_2| = 2 \Rightarrow y = 201 \pm 2$$

$$y = x_1 \cdot x_2 = 10100, \quad \left| \frac{\Delta y}{y} \right| \leq \left| \frac{\Delta x_1}{x_1} \right| + \left| \frac{\Delta x_2}{x_2} \right|$$

$$= \frac{1}{101} + \frac{1}{100} = 0.019901\dots$$

$$\Rightarrow |\Delta y| = 10100 \cdot 0.019901\dots = 201 \Rightarrow y = 10100 \pm 201$$

\therefore 1% respektive 2% maxalfel i beräkningsresultatet.

Kancellation

Vid subtraktion av två nästan lika stora tal x_1 och x_2 uppstår en noggrannhetsförlust som kallas **kancellation**.

Exempel 5

Ekvationen $x^2 - 20x + 1 = 0$ har rötterna $x_{1,2} = 10 \pm \sqrt{99}$. Antag att vi känner $\sqrt{99}$ med 4 korrekta decimaler. Hur noggrant kan rötterna bestämmas?

$$x_1 = 10 + \sqrt{99} = 10 + 9.9499 \pm 0.5 \cdot 10^{-4} = 19.9499 \pm 0.5 \cdot 10^{-4}$$

$$x_2 = 10 - \sqrt{99} = 10 - 9.9499 \pm 0.5 \cdot 10^{-4} = 0.0501 \pm 0.5 \cdot 10^{-4}$$

∴ 6 respektive 3 (!) signifikanta siffror.

Förlängning med konjugatet undviker cancellation:

$$x_2 = \frac{(10 - \sqrt{99})(10 + \sqrt{99})}{(10 + \sqrt{99})} = \frac{1}{10 + \sqrt{99}} = \frac{1}{19.9499 \pm 0.5 \cdot 10^{-4}}$$

Exempel 5 (forts)

Vi har

$$\frac{1}{19.9499} = 0.050125564...$$

Regeln för felfortplantning vid division ger relativa felet ($\epsilon_r = \frac{|\Delta y|}{|y|}$) i närmvärdet till x_2 som kommer från felet i närmvärdet till $\sqrt{99}$:

$$\epsilon_r = \frac{0.5 \cdot 10^{-4}}{19.9499} = 2.50627... \cdot 10^{-6} < 0.3 \cdot 10^{-5}$$

Absoluta felet:

$$\epsilon_r \cdot |x_2| < 0.3 \cdot 10^{-5} \cdot 0.050125564... = 1.50376... \cdot 10^{-7} < 0.5 \cdot 10^{-6}$$

$$\Rightarrow x_2 = 0.050125 \pm 0.5 \cdot 10^{-6}$$

6 korrekta decimaler och 5 signifikanta siffror!

Exempel 6

Vi har

$$f(x) = \frac{1 - \cos x}{\sin x} = \frac{(1 - \cos x)(1 + \cos x)}{\sin x(1 + \cos x)} = \frac{\sin^2 x}{\sin x(1 + \cos x)} = \frac{\sin x}{1 + \cos x}$$

För små x är $\cos x \approx 1$ och vi får cancellation i det första uttrycket men inte i det sista. Undvik cancellation genom omskrivning.

Matematiskt ekvivalenta uttryck är inte alltid numeriskt ekvivalenta!

Kancellation kan ofta undvikas med Taylorutveckling. För små x är t ex:

$$1 - \cos x \approx 1 - \left(1 - \frac{x^2}{2} + \frac{x^4}{4!}\right) = \frac{x^2}{2} - \frac{x^4}{4!}$$

Talrepresentation i datorer

Heltal kan representeras exakt i en dator (om ordlängden räcker till).

Hur ska vi representera godtyckliga reella tal?

- Ett reellt tal kan skrivas som ett **flyttal** (dvs på exponentiell form):

$$763.45 = 7.6345 \cdot 10^2$$

$$\frac{\pi}{16} = 0.1963495408493620... = 1.963495408493620... \cdot 10^{-1}$$

- Ett flyttal kallas normaliserat om det endast finns en siffra ($\neq 0$) framför decimalpunkten.
- Talet $7.6345 \cdot 10^2$ har heltalsdelen 7 och bråkdelen 0.6345. Detta betyder

$$(7 \cdot 10^0 + 6 \cdot 10^{-1} + 3 \cdot 10^{-2} + 4 \cdot 10^{-3} + 5 \cdot 10^{-4}) \cdot 10^2$$

- Vi har alltså ett positionssystem med basen 10.

Hur ser datorns flyttalssystem ut?

Talrepresentation i datorer

Definition 4

Ett flyttalssystem karakteriseras av parametrar (β, t, L, U) , där β är talsystemets bas, t är antalet siffror i bråkdelen, och L och U är systemets minsta respektive största exponent.

Exempel 7

Talsystemet $(10, 3, -9, 9)$ innehåller exempelvis talen

$$4.562 = 4.562 \cdot 10^0, \quad 123.7 = 1.237 \cdot 10^2 \quad \text{och} \quad 0.006532 = 6.532 \cdot 10^{-3}.$$

Normaliserade flyttal

Varje reellt tal $x \neq 0$ avkortat till $t + 1$ siffror kan framställas i basen β på formen

$$x = \pm r \cdot \beta^N$$

där $r = d_0.d_1d_2\dots d_t$, $1 \leq d_0 < \beta$, kallas **taldelen** (eller **mantissan**) och heltalet N **exponentdelen**.

OBS! Talet 0 kan inte skrivas som ett normaliserat flyttal.

Talrepresentation i datorer

IEEE 754 Binary32 (Enkel precision): (2, 23, -126, 127)

I datorn lagras talet som ett ord (32 bitar). Bitarna fördelas enligt

| | | |
|-----------|-------------|--------------|
| s (1 bit) | e (8 bitar) | f (23 bitar) |
|-----------|-------------|--------------|

- I normalfallet, $1 \leq e \leq 254$, gäller att flyttalet skall tolkas som

$$x = (-1)^s (1.f)_2 \cdot 2^{e-127}$$

- $e = 0$ eller $e = 255$ ger möjlighet att definiera $x = 0$, $x = \pm\infty$, och $x = \text{NaN}$.
- Systemet kan representera tal mellan approximativt 10^{-40} och 10^{40} .

Exempel 8

Hur lagras talet 12.39 i datorn?

$$(12.39)_{10} = 1.54875 \cdot 2^3 = (-1)^0 (1.10001100011110101110000)_2 \cdot 2^{130-127}$$

$$\Rightarrow s = 0, e = (130)_{10} = (10000010)_2, f = (10001100011110101110000)_2$$



Talrepresentation i datorer

Maskinepsilon

Ett normaliserat flyttal med precision p representeras i basen 2 som

$$x = \pm r \cdot 2^N = (1.f_1f_2, \dots, f_{p-1}) \cdot 2^N$$

Om vi ökar mantissan r med 1 i den sista biten får vi nästa representerbara tal:

$$x_{\text{nästa}} = (\pm r + 2^{-(p-1)})2^N$$

$$\Rightarrow x_{\text{nästa}} - x = 2^{-(p-1)} \cdot 2^N \Leftrightarrow \frac{x_{\text{nästa}} - x}{|x|} = \frac{2^{-(p-1)}2^N}{r2^N} = \frac{2^{-(p-1)}}{r} \leq 2^{-(p-1)}$$

För ett flyttal x med minsta möjliga mantissa ($r = 1.000\dots$) får vi

$$x_{\text{nästa}} - x = 2^{-(p-1)}|x| = \varepsilon|x|$$

Om $x = 1$ är $1_{\text{nästa}} = 1 + \varepsilon$ dvs ε är avståndet mellan 1 och närmast större flyttal.

Definition

Maskinepsilon (ε) är det minsta positiva talet sådant att $1 + \varepsilon \neq 1$ i flyttalsaritmetik.

ε anger den **största relativa skillnaden mellan två intelligande flyttal** och mäter alltså den **relativa noggrannheten** i flyttalsrepresentationen.

Talrepresentation i datorer

Maskinepsilon

- IEEE 754 Binary32 (Enkel precision): $\varepsilon = 2^{-23} \approx 1.19 \times 10^{-7}$.
- IEEE 754 Binary64 (Dubbel precision): $\varepsilon = 2^{-52} \approx 2.22 \times 10^{-16}$.

Vid numeriska beräkningar: Maskinepsilon (ε)

- Bestämmer den **noggrannhetsgräns** som beräkningarna kan uppnå.
- Orsakar **avrundningsfel** vid aritmetiska operationer.
- Påverkar **stabiliteten** hos algoritmer - särskilt vid subtraktion av nästan lika stora tal (kancellation).
- Påverkar valet av **toleranser** i iterativa metoder (t.ex. stoppkriterier).

Anm:

- Flyttal är alltså **ojämnt fördelade längs tallinjen**: Nära 0 ligger talen närmare, längre bort glesare.

Talrepresentation i datorer

OBS! Då vi lagrar $x = 0.1$ i flyttalssystemet $(2, 23, -126, 127)$ får vi

$$\begin{aligned} x = (0.1)_{10} &= (0.00011001100110011001100110011001100...)_2 \\ &= (1.10011001100110011001100110011001100...)_2 \cdot 2^{-4} \end{aligned}$$

Med 23 bitar i bråkdelen blir inte $x = 0.1$ lagrat exakt i datorn.

Vi får ett avrundningsfel:

$$|x^* - x| \leq 2^{-27} = 7.45 \cdot 10^{-9}$$

Är det viktigt?

Ett tal som kan lagras exakt i det decimala talsystemet kan inte säkert lagras exakt i det binära.

Felen är små men datorer kan göra många beräkningar snabbt.

Anm: Reella tal kan inte alltid representeras exakt i en dator:

$$\sqrt{2} = 1.4142135623... \quad \leftarrow \text{irrationellt tal med oändligt antal decimaler}$$

- Matematiskt: $\sqrt{2} \cdot \sqrt{2} = 2$
- I datorn: $\sqrt{2} \cdot \sqrt{2} \neq 2$